Final Project Presentation

Nomor Kelompok: 5
Nama Mentor: Ramdhan Hidayat
Nama:

- <Juwita Natalia Sinaga>
- <Rima Chusnul Magfiroh>

Machine Learning Class

Program Studi Independen Bersertifikat
Zenius Bersama Kampus Merdeka

Petunjuk

- Waktu presentasi adalah 5 menit (tentatif, tergantung dari banyaknya kelompok yang mendaftarkan diri)
- Waktu tanya jawab adalah 5 menit
- Silakan menambahkan gambar/visualisasi pada slide presentasi
- Upayakan agar tetap dalam format poin-poin (ingat, ini presentasi, bukan esai)
- Jangan masukkan *code* ke dalam slide presentasi (tidak usah memasukan screenshot jupyter notebook)

Latar Belakang

Latar Belakang Project

Sumber Data:
https://www.kaggle.com/datasets/yasserh/loan-default-dataset

Problem: **classification**

Tujuan:

- memprediksi status loan berdasarkan faktor-faktor yang diperhatikan perbankan agar mereka tidak salah dalam memberikan *loan* kepada nasabah.

Explorasi Data dan Visualisasi

Business Understanding

Menganalisis dataset *loan* (pinjaman perbankan) yang menyimpan data historis nasabah bank yang cenderung default (gagal bayar pinjaman) atau tidak. Mengidentifikasi nasabah yang berisiko tinggi untuk gagal bayar adalah salah satu cara untuk meminimalisir kerugian pemberi pinjaman. Untuk itu, kita akan coba memprediksi kemungkinan nasabah gagal bayar menggunakan prediktor-prediktor yang disediakan. Target kolomnya adalah ‘Status’ dengan keterangan 0 ditolak dan 1 diterima.

Data Cleansing

Dataset Loan default memiliki 34 kolom dan 148670 baris. Dengan kolomnya adalah

```
df.columns
```

```
Index(['ID', 'year', 'loan_limit', 'Gender', 'approv_in_adv', 'loan_type',
       'loan_purpose', 'Credit_Worthiness', 'open_credit',
       'business_or_commercial', 'loan_amount', 'rate_of_interest',
       'Interest_rate_spread', 'Upfront_charges', 'term', 'Neg_ammortization',
       'interest_only', 'lump_sum_payment', 'property_value',
       'construction_type', 'occupancy_type', 'Secured_by', 'total_units',
       'income', 'credit_type', 'Credit_Score', 'co-applicant_credit_type',
       'age', 'submission_of_application', 'LTV', 'Region', 'Security_Type',
       'Status', 'dtir1'],
      dtype='object')
```

Data Cleansing

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 148670 entries, 0 to 148669
Data columns (total 34 columns):
 #   Column                     Non-Null Count   Dtype
---  ------                     --------------   -----
 0   ID                         148670 non-null  int64
 1   year                       148670 non-null  int64
 2   loan_limit                 145326 non-null  object
 3   Gender                     148670 non-null  object
 4   approv_in_adv              147762 non-null  object
 5   loan_type                  148670 non-null  object
 6   loan_purpose               148536 non-null  object
 7   Credit_Worthiness          148670 non-null  object
 8   open_credit                148670 non-null  object
 9   business_or_commercial     148670 non-null  object
 10  loan_amount                148670 non-null  int64
 11  rate_of_interest           112231 non-null  float64
 12  Interest_rate_spread       112031 non-null  float64
 13  Upfront_charges            109028 non-null  float64
 14  term                       148629 non-null  float64
 15  Neg_ammortization          148549 non-null  object
 16  interest_only              148670 non-null  object
 17  lump_sum_payment           148670 non-null  object
 18  property_value             133572 non-null  float64
 19  construction_type          148670 non-null  object
 20  occupancy_type             148670 non-null  object
 21  Secured_by                 148670 non-null  object
 22  total_units                148670 non-null  object
 23  income                     139520 non-null  float64
 24  credit_type                148670 non-null  object
 25  Credit_Score               148670 non-null  int64
 26  co-applicant_credit_type   148670 non-null  object
 27  age                        148470 non-null  object
 28  submission_of_application  148470 non-null  object
 29  LTV                        133572 non-null  float64
 30  Region                     148670 non-null  object
 31  Security_Type              148670 non-null  object
 32  Status                     148670 non-null  int64
 33  dtir1                      124549 non-null  float64
dtypes: float64(8), int64(5), object(21)
memory usage: 38.6+ MB
```

Data Cleansing

```
df["total_units"].value_counts()
```

```
1U    146480
2U      1477
3U       393
4U       320
Name: total_units, dtype: int64
```

Terdapat kejanggalan pada tipe data kolom total_units dan age yang memiliki tipe data object, padahal total unit dan usia seharusnya bertipe data integer. Ternyata, kolom total_units dikelompokkan berdasarkan jumlah unitnya dan kolom age dikelompokkan berdasarkan range usia tertentu sehingga kedua kolom ini memiliki tipe data object. Nilai-nilai pada total_units akan diubah menjadi integer.

```
df["Gender"].value_counts()
```

```
Male                 42346
Joint                41399
Sex Not Available    37659
Female               27266
Name: Gender, dtype: int64
```

Kolom gender memiliki nilai 'Sex Not Available' yang sama dengan nilai NaN, maka dari itu nilai 'Sex Not Available akan diganti dengan NaN'

Data Cleansing

```
Data columns (total 34 columns):
 #   Column                     Non-Null Count   Dtype
---  ------                     --------------   -----
 0   ID                         148670 non-null  int64
 1   year                       148670 non-null  int64
 2   loan_limit                 145326 non-null  object
 3   Gender                     111011 non-null  object
 4   approv_in_adv              147762 non-null  object
 5   loan_type                  148670 non-null  object
 6   loan_purpose               148536 non-null  object
 7   Credit_Worthiness          148670 non-null  object
 8   open_credit                148670 non-null  object
 9   business_or_commercial     148670 non-null  object
 10  loan_amount                148670 non-null  int64
 11  rate_of_interest           112231 non-null  float64
 12  Interest_rate_spread       112031 non-null  float64
 13  Upfront_charges            109028 non-null  float64
 14  term                       148629 non-null  float64
 15  Neg_ammortization          148549 non-null  object
 16  interest_only              148670 non-null  object
 17  lump_sum_payment           148670 non-null  object
 18  property_value             133572 non-null  float64
 19  construction_type          148670 non-null  object
 20  occupancy_type             148670 non-null  object
 21  Secured_by                 148670 non-null  object
 22  total_units                148670 non-null  int64
 23  income                     139520 non-null  float64
 24  credit_type                148670 non-null  object
 25  Credit_Score               148670 non-null  int64
 26  co-applicant_credit_type   148670 non-null  object
 27  age                        148470 non-null  object
 28  submission_of_application  148470 non-null  object
 29  LTV                        133572 non-null  float64
 30  Region                     148670 non-null  object
 31  Security_Type              148670 non-null  object
 32  Status                     148670 non-null  int64
 33  dtir1                      124549 non-null  float64
dtypes: float64(8), int64(6), object(20)
memory usage: 38.6+ MB
```

Data Cleansing

```
df.Gender.isnull().groupby([df['Status']]).sum().astype(int)
```

```
Status
0    26892
1    10767
Name: Gender, dtype: int32
```

```
df.rate_of_interest.isnull().groupby([df['Status']]).sum().astype(int)
```

```
Status
0        0
1    36439
Name: rate_of_interest, dtype: int32
```

```
df.Interest_rate_spread.isnull().groupby([df['Status']]).sum().astype(int)
```

```
Status
0        0
1    36639
Name: Interest_rate_spread, dtype: int32
```

```
df.Upfront_charges.isnull().groupby([df['Status']]).sum().astype(int)
```

```
Status
0     3156
1    36486
Name: Upfront_charges, dtype: int32
```

Dataset df memiliki banyak kolom yang memiliki nilai kosong (missing value), terutama di kolom Gender, rate_of_interest, Interest_rate_spread, dan juga Upfront_charges. Akan dilihat persebaran missing value ini berdasarkan Statusnya.

Kolom rate_of_interest, Interest_rate_spread, dan Upfront_charges akan dihapus karena nilai kosong pada Status 1-nya sangat banyak. Begitu pula dengan kolom gender akan dihapus.

Data Cleansing

Kolom ID tidak berpengaruh terhadap Status, begitu pula dengan kolom year karena hanya memiliki satu nilai yaitu 2019. Maka dari itu, kolom ID dan year akan dihapus.

```
df.drop(['ID','year'], axis = 'columns', inplace = True)
```

Selanjutnya, data yang kosong akan diganti dengan mean/modus dari tiap kolomnya.

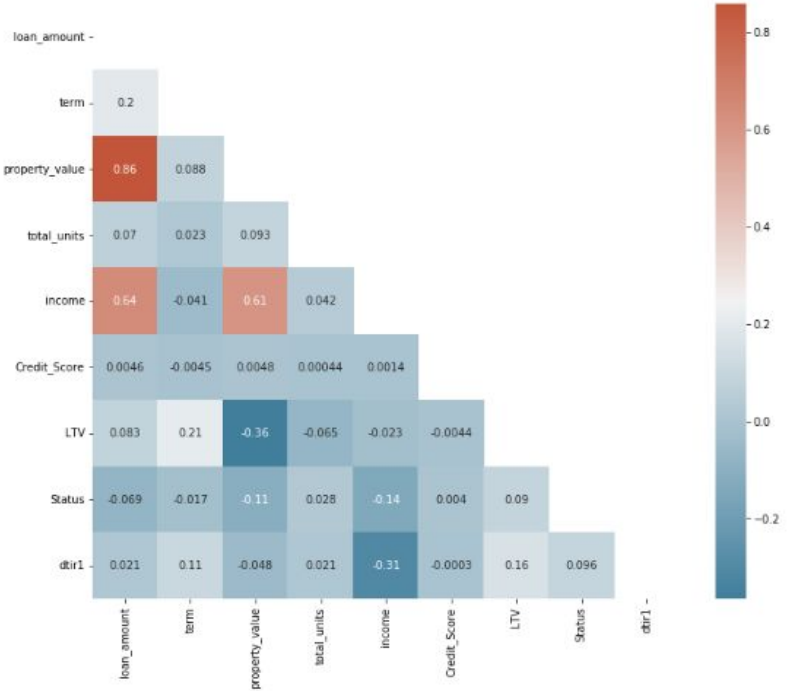
```
data = df.copy()
```

```
df['loan_limit'] = df['loan_limit'].fillna(df['loan_limit'].mode())
df['approv_in_adv'] = df['approv_in_adv'].fillna(df['approv_in_adv'].mode())
df['loan_purpose'] = df['loan_purpose'].fillna(df['loan_purpose'].mode())
df['term'] = df['term'].fillna(df['term'].mean())
df['Neg_ammortization'] = df['Neg_ammortization'].fillna(df['Neg_ammortization'].mode())
df['property_value'] = df['property_value'].fillna(df['property_value'].mean())
df['income'] = df['income'].fillna(df['income'].mean())
df['LTV'] = df['LTV'].fillna(df['LTV'].mean())
df['dtir1'] = df['dtir1'].fillna(df['dtir1'].mean())
```

Exploratory Data Analysis and Visualization

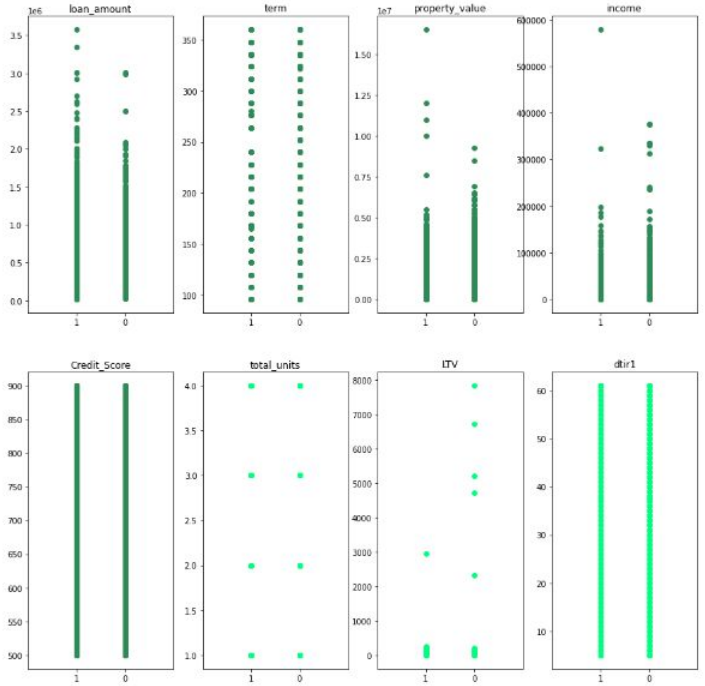
Sebelum melakukan feature selection, menentukan kolom yang akan digunakan sebagai prediktor. Disamping adalah heatmap korelasi antar tiap kolom numerik.

Disimpulkan bahwa property_value dan loan_amount memiliki korelasi yang tinggi. Sehingga jika salah satu dipilih menjadi prediktor, maka yang lainnya harus diserakan. Begitu pula kolom income dengan loan_amount dan kolom income dengan property_value memiliki korelasi yang cukup tinggi. Semakin tinggi property value-nya, semakin tinggi income dan juga loan amountnya, begitu pula sebaliknya.

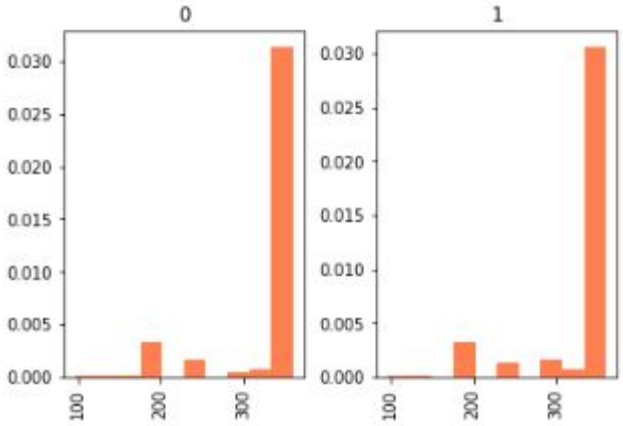


Exploratory Data Analysis and Visualization

Akan dilihat hubungan antar kolom numerik dengan kolom Status dengan menggunakan scatterplot. Dari visualisasi, pinjaman yang statusnya diterima jumlah maksimalnya justru lebih tinggi dibandingkan dengan yang statusnya ditolak, begitu pula dengan jumlah minimalnya lebih rendah. Selain itu, loan_amount yang tinggi juga mengindikasikan bahwa si peminjam punya income yang lebih tinggi, begitu pula sebaliknya, ditunjukkan dengan property_value dan income yang memiliki korelasi yang cukup tinggi dengan loan_amount.



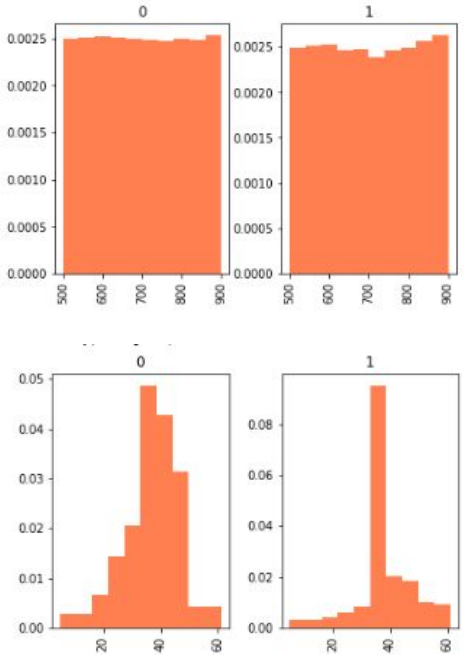
Exploratory Data Analysis and Visualization



Pada visualisasi, terlihat bahwa baik di status 0 maupun 1, semakin panjang term-nya, semakin banyak aplikasi pinjamannya. Hal ini mungkin dikarenakan jangka waktu pinjaman yang panjang membuat pembayaran tiap bulannya lebih rendah.

Dicek dengan groupby describe, kolom LTV dan total_units juga tidak begitu berpengaruh terhadap Status sehingga kolom tersebut tidak akan dipakai sebagai prediktor. Terakhir, kolom term, Credit_Score dan dtir1 memiliki rata-rata sedikit lebih tinggi pada pinjaman yang diterima. Berikut ditampilkan histogram Credit_Score dan dtir1 berdasarkan statusnya.

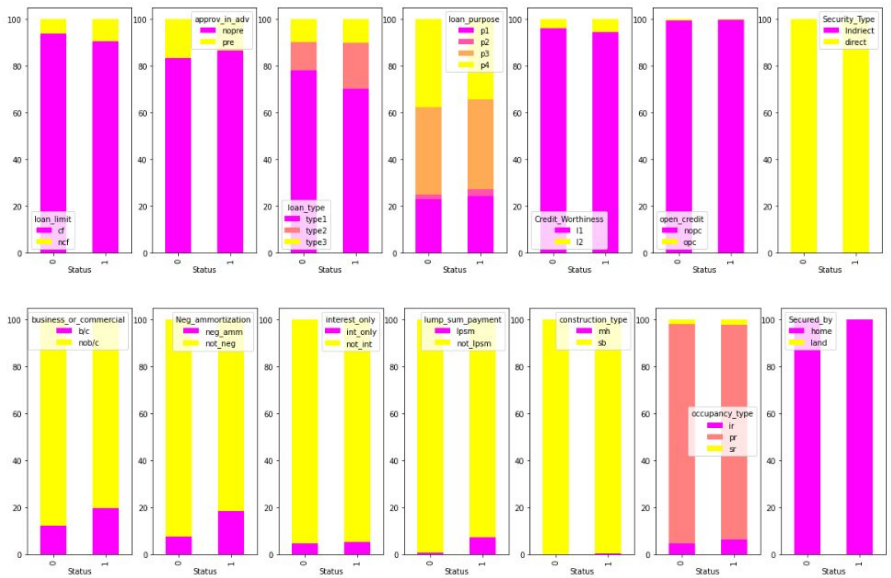
Exploratory Data Analysis and Visualization

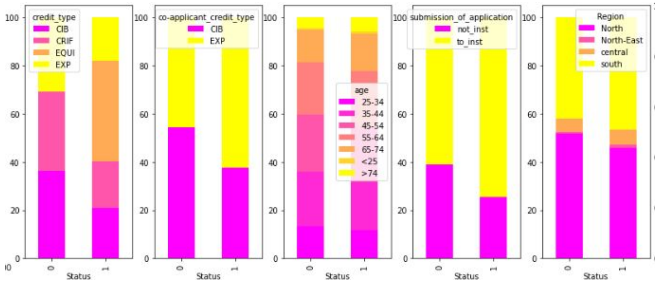


Histogram atas adalah credit_score, dimana pada status 1 cenderung naik seiring bertambahnya score, hal ini karena credit_score dihasilkan dari riwayat yang menunjukkan kredibilitas si peminjam. Histogram bawah adalah dtr1, range 40-60 pada status 1 lebih banyak presentasenya dibandingkan dengan status 0. Yang berarti semakin tinggi dtir, kemungkinan peminjam untuk mendapat pinjaman lebih tinggi.

Exploratory Data Analysis and Visualization

Selanjutnya akan dipilih kolom-kolom categorical yang akan menjadi feature dengan menggunakan barplot.





Exploratory Data Analysis and Visualization

Pada barplot, dapat dilihat dengan jelas bahwa kolom open_credit, interest_only, construction_type, occupancy_type, Secured_by, credit_type, co-applicant_credit_type, region, dan Security_Type tidak memiliki perbedaan jumlah yang signifikan antara pinjaman yang diterima dan tidak. Maka dari itu, kolom-kolom tersebut akan dihapus dan tidak digunakan sebagai prediktor.

age	25-34	35-44	45-54	55-64	65-74	<25	>74
Status							
0	13.294535	22.769591	23.539020	21.522614	13.543573	0.847980	4.482688
1	11.657839	20.058179	22.912264	23.112599	15.288564	1.062049	5.908505

Pada kolom age dapat dilihat bahwa peminjam dengan usia dibawah 54 cenderung lebih banyak ditolak dibandingkan dengan peminjam diatas usia 54.

loan_type	type1	type2	type3
Status			
0	78.012336	12.130571	9.857093
1	70.348536	19.574770	10.076694

Peminjam yang diapprove memiliki persentase yang lebih tinggi pada tipe 2 dan 3 nya dibandingkan dengan yang tidak diapprove. Pada model ini, kolom age dan loan_type akan digunakan sebagai prediktor. Kolom lainnya yang tidak digunakan akan dihapus

Exploratory Data Analysis and Visualization

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 148670 entries, 0 to 148669
Data columns (total 9 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   loan_type       148670 non-null  object
 1   loan_amount     148670 non-null  int64
 2   term            148670 non-null  float64
 3   property_value  148670 non-null  float64
 4   income          148670 non-null  float64
 5   Credit_Score    148670 non-null  int64
 6   age             148470 non-null  object
 7   Status          148670 non-null  int64
 8   dtir1           148670 non-null  float64
dtypes: float64(4), int64(3), object(2)
memory usage: 10.2+ MB
```

Feature yang digunakan adalah loan_type, loan_amount, term, property_value, income, Credit_Score, age, dan dtir1.

```
df2['Status'].value_counts()
```

```
0    112031
1     36639
Name: Status, dtype: int64
```

Kolom status memiliki data yang tidak seimbang. Oleh karena itu, akan dilakukan resampling terlebih dahulu.

Modelling

Resampling & One Hot Encoding

Sebelum melakukan one-hot encoding, akan dilakukan resampling data terlebih dahulu karena terdapat perbedaan banyaknya data yang disetujui dan ditolak pada kolom Status. Menggunakan metode downsampling.

```
from sklearn.utils import resample

#create two different dataframe of majority and minority class
df_majority = df2[(df2['Status']==0)]
df_minority = df2[(df2['Status']==1)]
# upsample minority class
df_majority_downsampled = resample(df_majority,
                                   replace=True,    # sample with replacement
                                   n_samples=36639, # to match majority class
                                   random_state=42)  # reproducible results
# Combine majority class with upsampled minority class
df_downsampled = pd.concat([df_majority_downsampled, df_minority])
```

```
df_downsampled['Status'].value_counts()
```

```
0    36639
1    36639
Name: Status, dtype: int64
```

Random Forest

```
X = df2.loc[:, df2.columns != 'Status']
y = df2["Status"]
```

```
# Train-test-split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,
                                                    random_state=1)
```

```
from sklearn.ensemble import RandomForestClassifier

classifier_rf = RandomForestClassifier(random_state=42, n_jobs=-1, max_depth=5,
                                       n_estimators=100, oob_score=True)

classifier_rf.fit(X_train, y_train)
```

```
RandomForestClassifier(max_depth=5, n_jobs=-1, oob_score=True, random_state=42)
```

Improving Model

Pada tahap ini, akan digunakan dua cara untuk meningkatkan akurasi model, yaitu dengan cara menambahkan feature yang digunakan sebagai prediktor dan dengan hyperparameter tuning.

Adding feature

```
df3 = df.copy()
```

```
df3.drop(['lump_sum_payment','Neg_ammortization','business_or_commercial','approv_in_adv','loan_limit','Credit_Worthiness','submission_of_application'
```

```
from sklearn.utils import resample

#create two different dataframe of majority and minority class
df_majority = df3[(df3['Status']==0)]
df_minority = df3[(df3['Status']==1)]
# upsample minority class
df_majority_downsampled = resample(df_majority,
                                   replace=True,    # sample with replacement
                                   n_samples=36639, # to match majority class
                                   random_state=42)  # reproducible results
# Combine majority class with upsampled minority class
df_downsampled = pd.concat([df_majority_downsampled, df_minority])
```

Improving Model

```
X = df3.loc[:, df3.columns != 'Status']
y = df3["Status"]
```

```
from sklearn.model_selection import train_test_split
X_train2, X_test2, y_train2, y_test2 = train_test_split(X, y, test_size=0.4,
                                                        random_state=1)
```

```
from sklearn.ensemble import RandomForestClassifier

improve_rf = RandomForestClassifier(random_state=42, n_jobs=-1, max_depth=5,
                                    n_estimators=100, oob_score=True)

improve_rf.fit(X_train2, y_train2)
```

```
RandomForestClassifier(max_depth=5, n_jobs=-1, oob_score=True, random_state=42)
```

Hyperparameter Tuning

```
#Hyperparameter Tuning
#Mendefinisikan RandomForestClassifier
rf = RandomForestClassifier(random_state=42, n_jobs=-1)
```

```
#Mendefinisikan hyperparameter
params = {
    'max_depth': list(range(2,20)),
    'min_samples_leaf': list(range(1,50)),
    'n_estimators': list(range(100,200,10))
}
```

```
#random search untuk mencari hyperparameter terbaik
from sklearn.model_selection import RandomizedSearchCV

random_search = RandomizedSearchCV(estimator=rf,
                           param_distributions=params,
                           n_iter=100,
                           cv = 4,
                           n_jobs=-1, verbose=1, scoring="accuracy")

random_search.fit(X_train, y_train)
```

```
random_search.best_score_
```

```
0.7790111073410997
```

```
#train model dengan menggunakan hyperparameter hasil random search
rf_best = random_search.best_estimator_
rf_best.fit(X_train, y_train)
```

```
RandomForestClassifier(max_depth=19, min_samples_leaf=2, n_estimators=160,
                       n_jobs=-1, random_state=42)
```

```
#model awal
y_classifier_rf = classifier_rf.predict(X_test)
#setelah menambah feature
y_improve_rf = improve_rf.predict(X_test2)
#setelah hyperparameter tuning
y_rf = rf_best.predict(X_test)
```

Model Evaluation

```
#Evaluasi model menggunakan AUC
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_classifier_rf, pos_label=1) # pos_label: positive label
print(auc(fpr, tpr))
fpr, tpr, thresholds = roc_curve(y_test2, y_improve_rf, pos_label=1) # pos_label: positive label
print(auc(fpr, tpr))
fpr, tpr, thresholds = roc_curve(y_test, y_rf, pos_label=1) # pos_label: positive label
print(auc(fpr, tpr))
```

```
0.7635607160248983
0.7575338356203899
0.7829313038637428
```

```
#Evaluasi Model menggunakan classification report
from sklearn.metrics import classification_report
print(classification_report(y_test, y_classifier_rf))
print(classification_report(y_test2, y_improve_rf))
print(classification_report(y_test, y_rf))
```

```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, y_classifier_rf))
print(accuracy_score(y_test2, y_improve_rf))
print(accuracy_score(y_test, y_rf))
```

```
0.7644650655021834
0.7581195414847162
0.7837745633187773
```

```
print(accuracy_score(y_test, y_rf)-accuracy_score(y_test, y_classifier_rf))
```

```
0.019309497816593857
```

```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, y_classifier_rf))
print(confusion_matrix(y_test2, y_improve_rf))
print(confusion_matrix(y_test, y_rf))
```

```
[[13261  1493]
 [ 5411  9147]]
[[12469  2285]
 [ 4805  9753]]
[[13412  1342]
 [ 4996  9562]]
```

Menambahkan feature loan_purpose ternyata tidak menambahkan akurasi model, model awal lebih baik daripada model setelah ditambahkan feature loan_purpose. Hyperparameter tuning meningkatkan akurasi model sebesar 1.93%. Model setelah hyperparameter tuning memiliki nilai presisi 80% dan recall 78%.

Conclusion

- Akurasi model

  Awal (metode random forest) : 76.44%

  Setelah menambahkan feature : 75.81%

  Setelah hyperparameter tuning : 78.37%
- Feature-feature penting : `loan_type`, `loan_amount`, `term`, `property_value`, `income`, `Credit_Score`, `age`, dan `dtir1`.
- Peminjam pada usia>55 tahun lebih banyak disetujui dibandingkan dengan peminjam pada usia < 55 tahun.
- `loan_type` berpengaruh dimana loan_type type 2 lebih banyak disetujui.

- Peminjam lebih memilih jangka waktu pinjaman yang panjang (term), hal ini dikarenakan jangka waktu yang panjang membuat cicilan per bulannya lebih rendah.
- Credit_Score seringkali digunakan dalam menentukan apakah suatu pinjaman disetujui atau tidak. Namun ternyata, credit_score yang tinggi tidak menjamin pinjaman akan disetujui.
- loan_amount, income, dan property_value memiliki korelasi yang cukup tinggi. Sehingga loan_amount saja tidak bisa memprediksi apakah pinjaman akan disetujui atau tidak, income dan property_value perlu disertakan.
- Saran kami untuk perusahaan pinjaman yaitu untuk tidak hanya memperhatikan Credit_Score saja, namun juga faktor lain yang berpengaruh seperti term, loan_amount,loan_type, property_value, income, age, dan juga dtir1.

Terima kasih!

Ada pertanyaan?